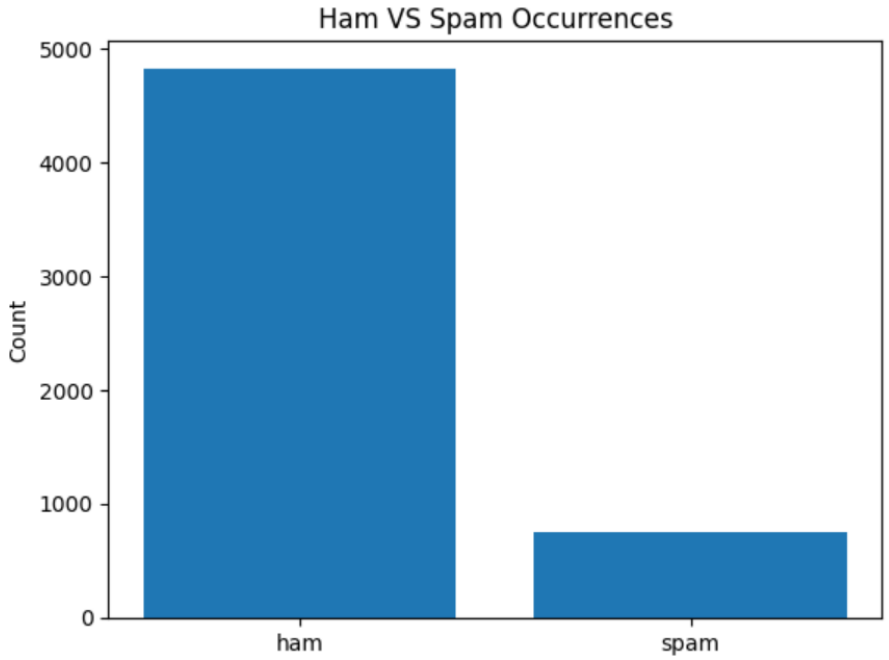
Spam Email Detection with Machine Learning

- **Dataset :** SMS Spam Collection Dataset
- This project is **the 1st task of Code Casa Data science internship**.
- By developing this detector, we aim to enhance email security and protect users from falling victim to spam emails.
- 2 labels : **HAM , SPAM**

1- Data Loading and visualize it:

After loading data , I noted that :

- 99% of unnamed records in the last 3 columns are null values so I dropped them.
- the data is imbalanced:



I applied label Encoding using label_encoder object that knows how to understand word labels.

2- Text processing :

Building a machine learning model requires the preprocessing of data, and the quality of the preprocessing determines the model's performance.

Where preprocessing text is the initial stage in NLP's model-building process.

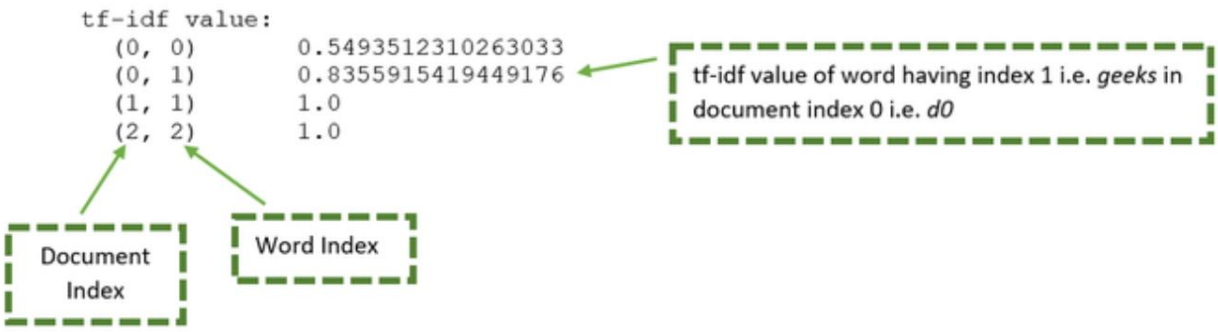
- **Remove Punctuation using translate** function.
- **Tokenization.**
- **Removing Stopwords.**
- **Remove all characters other than the alphabet.**
- **Part of Speech Tagging** : I used Statistical POS tagging(spaCy library) as it's more accurate than rule-based taggers.

3- **Loud Words in Spam Email:**
generated a word cloud visualization based on the provided text data, where more frequently occurring words are displayed in larger font sizes.

4- **Feature extraction using TF-IDF:**
It is primarily used to measure the importance of a term within a document or a corpus of documents. While TF-IDF is often applied at the lexical level, it is not limited to it and can be used at different levels of analysis.

But make sure that after fitting the vectorizer on the training data, you need to apply the same transformation to the test data using the transform method of the vectorizer. This will ensure that the test data is represented using the same features as the training data.

This example :



5- **Balancing the Dataset using SMOTE algorithm:**
I chose to use **SMOTE algorithm** where it is an oversampling method to solve the imbalance problem. It aims to balance class distribution by randomly increasing minority class examples by replicating them.
The SMOTE algorithm requires numerical input, so I used tf-idf results of training data.

The following image show you that data became balanced.

```
print((y_train_res == 1).sum())
print((y_train_res == 0).sum())
```

```
3391
3391
```

it is not necessary to balance the test data as well.

The purpose of balancing the training data is to help the model learn from both classes more effectively and avoid biases towards the majority class. By oversampling the minority class.

6- Models Development:

This is the conclusion about the models and its accuracy with balanced and imbalanced data where the Support Vector Machine (SVM) model demonstrated exceptional performance on both the imbalanced and balanced datasets, achieving an impressive accuracy rate of approximately 97%.

	Balanced Accuracy	Imbalanced Accuracy
Random Forest	0.97189	0.967703
Logistic Regression	0.97488	0.956340
SVM	0.97488	0.973684